Near-Eye Display Eye Tracking via Convolutional Neural Networks

Robert Konrad Shikhar Shrestha Paroma Varma

Introduction

Near-eye displays, such as virtual reality (VR) and augmented reality (AR) systems, are beginning to enter the consumer market. One cue that VR and AR displays are currently missing is depth of field rendering. The depth of field blur is a strong cue used for depth perception, but also a technique increases immersion and a “fun” factor in virtual reality if it can be implemented in a gaze-contingent manner in real-time[1,2]. Rendering a depth of field into a scene in real-time is not difficult as there has been much work done in this area for video games previously [3,4,5,6]. The main challenge that we attempt to address with this project is performing gaze-tracking in a scene, particularly for near-eye display.

Most gaze-tracking has been done on users looking at a screen from a distance. For head-mounted displays, a camera must be installed inside, without obstructing the view of the user while simultaneously being able to capture the full view of the pupil and iris, regardless of the gaze of the user. This has proven to be a challenging issue, as there is only one company, SMI, that is able to implement near-eye display eye tracking, at the cost of $10,000. We seek a more elegant solution to the problem that might be able to place a camera in a suboptimal position where it might not capture a full view of the eye’s pupil and iris. We will not use traditional gaze-tracking algorithms, but instead use a series of input/output pairs to train a neural network that will predict where a user is looking based on partial images of the user’s pupil and iris.

Our Proposal

A CNN based approach to eye tracking instead of feature based methods that are pervasive in the industry could be very powerful as it has the ability to account for variability in the images. The main challenge with eye tracking systems is that people have different skin tone, eye color, size and proportions and the FoV isn’t always perfectly aligned. This creates problems with feature based tracking methods as the detectors are often not able to detect feature points of the user’s eye.

The core hypothesis of this work is to use a CNN that has been trained on a large near eye display eye tracking dataset that we will capture ourselves and then use a brief training exercise to fine-tune the model for each user upon wearing the headset to calibrate and improve tracking accuracy. We propose to capture a large dataset by performing a user study where they follow a moving marker on a screen for 5 minutes. As the location of the marker is known, this results in 30X60X5 frames per user. The captured dataset will then be augmented using standard data augmentation methods and split into training, test and cross-validation sets. A small CNN such as the LeNet will then be trained to the dataset and we can iterate the model structure and hyperparameters using the cross-validation data. Caffe/TensorFlow would be used for training the model and inference which will then be ported to a small linux SBC to tracks the user’s gaze in real-time while he/she is wearing the near eye display. We will begin our experiments on a pre-defined dataset, to learn the models we should attempt to use and then we will create our own capture system with user calibration.

Completed Milestones

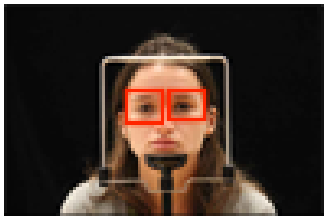
As stated above we first attempted to find a gaze-tracking dataset in the literature to begin prototyping of the CNN models and hyperparameters. A thorough list of datasets can be found in [7]. The datasets varied on various parameters: number of subjects, number of targets, number of head positions, whether each user was calibrated, and the resolution of the images. Upon reviewing the datasets we decided that for a near-eye display application, only one head position is necessary because the camera within the near-eye display will remain static relative to the eyes throughout use. Therefore a large number of head positions was not advantageous for us.

We are interested in a dataset with as many viewing targets as possible, so that our CNN can train at a fine resolution. Also, because we will initially use the CNN as a classification problem (it is the simplest first step), having a finer resolution of the targets is necessary. Eventually, we will create a CNN with regression in mind (which is the appropriate model for this sort of problem), to not have to quantize the image plane into a fixed number of bins.

With this two constraints in mind we decided on using the the Columbia Gaze Data Set, named CAVE. It has data from 56 subjects, with each one looking at 21 different targets, with 5 recorded head poses, for a total of 5880 images. The dataset is thorough, but has some limitations. Firstly, the dataset provides images of the full face of the user. We believe that some eyebox around the user's eyes contains the most information about where a user is looking, and we therefore want to extract those pixels from the full faced images. Our approach to this is explained below. Secondly, although the dataset provides 5880 images, if we remove the unnecessary head poses (we only need 1), we are left with 1176 images. This might not be enough to train a CNN on, and we therefore augment this dataset with noise additions and transformations, explained below.

Eye-box Detection

Because the full-faced images contain much information not relevant to our task of gaze-tracking, we attempt to extract only the information around each eye. We do this with the OpenCV library that implements the real-time gaze tracking algorithm described in "Accurate Eye Centre Localisation by Means of Gradients" [8]. This algorithm is used in multiple gaze tracking applications because of its ability to detect eyes in real-time (which is crucial for gaze-tracking purposes). You can see a sample of this algorithm being applied to the dataset below.

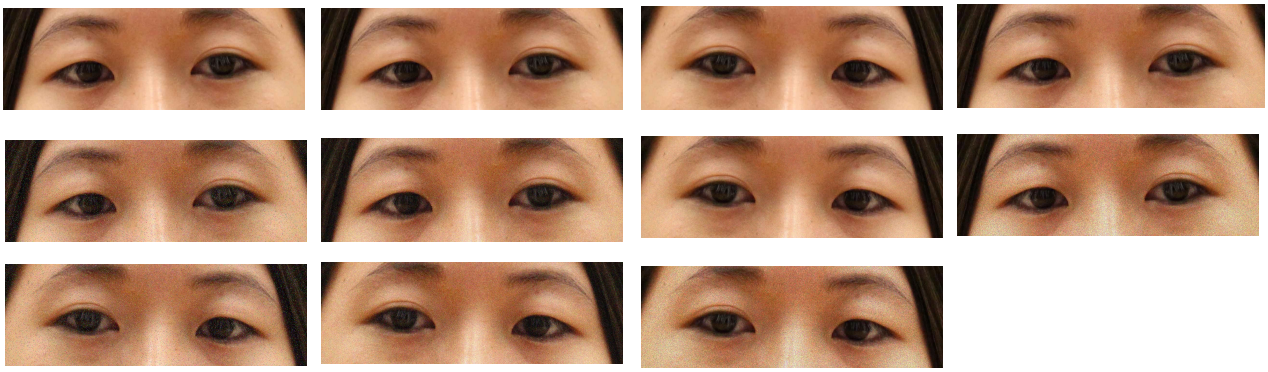


Dataset Augmentation

If we remove the unnecessary head poses, we are left with 1176 images for us to train on. Of course this is not enough for us to train a CNN for real-world environments, and we therefore augment the dataset.

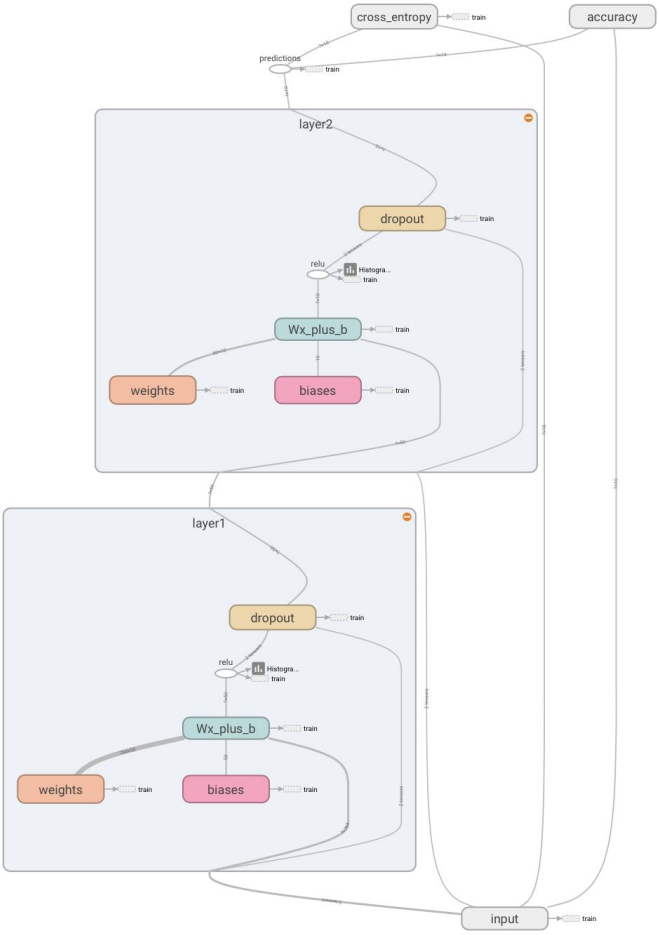
```
newimg1 = flipdim(newimg,2);
```

```
newimg2 = imnoise(newimg,'poisson');
newimg3 = imnoise(newimg,'salt & pepper');
newimg4 = imnoise(newimg,'gaussian');
newimg5 = imnoise(newimg,'speckle');
newimg6 = imnoise(newimg1,'poisson');
newimg7 = imnoise(newimg1,'salt & pepper');
newimg8 = imnoise(newimg1,'gaussian');
newimg9 = imnoise(newimg1,'speckle');
```



TensorFlow Setup

A simple image classification CNN was set up using TensorFlow to experiment with the different model architectures and data structures. One of the examples from the TensorFlow tutorials uses MNIST datasets, where each image has dimensions 28x28. In order to use a CNN on the eye-images, they need to be downsampled substantially. Using a simple downsampling algorithm can remove details in the eye images that are necessary to build a robust regression model. An additional CNN layer or two would be required to downsample the input data by converting it to lower resolution images or extracting canonical eye-detection features.



Remaining Milestones

The following milestones are left:

- Editing simple classification scheme to incorporate 21 viewpoints for above dataset (May 17th)
    - Downsampling given dataset properly
    - Training and Testing with given model
- Defining a regression CNN model (May 22nd)
    - Check accuracy with linear interpolation from 21 viewpoints?
    - Testing various architectures
- Building Camera Setup (May 20th)
    - Determining placement of camera for one vs two eye views Testing resolution of eye views captured when looking at screen/adjusting movement of the dot on the screen (May 19th)
- Collecting Real Data (May 23rd)
- CNN on Real Data (May 28th)
    - Converting real dataset to lmbd format
    - Training and testing on existing regression CNN

References

[1] R. Mantiuk, B. Bazyluk, and A. Tomaszewska. 2011. Gaze-Dependent depth-of-field effect rendering in virtual environments. In *Proceedings of the Second international conference on Serious Games Development and Applications* (SGDA'11). Springer-Verlag, Berlin, Heidelberg, 1-12.

[2] M. Mauderer, S. Conte, M. a. Nacenta, and D. Vishwanath, “Depth perception with gaze-contingent depth of field,” Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI ’14, pp. 217–226, 2014.

[3] B. a. Barsky and T. J. Kosloff, “Algorithms for Rendering Depth of Field Effects in Computer Graphics,” World Scientific and Engineering Academy and Society (WSEAS), pp. 999–1010, 2008.

[4] T. Zhou, J. X. Chen, and M. Pullen, “Accurate Depth of Field Simulation in Real Time,” Time, vol. 26, no. 1, pp. 15–23, 2007.

[5] C. Tomasi and R. Manduchi, “Bilateral filtering for gray and color images,” in Computer Vision, 1998. Sixth International Conference on, pp. 839–846, Jan 1998.

[6] S. Xu, X. Mei, W. Dong, X. Sun, X. Shen, and X. Zhang, “Depth of Field Rendering via Adaptive Recursive Filtering,” 2014.

[7] Onur Ferhat and Fernando Vilariño, “Low Cost Eye Tracking: The Current Panorama,” Computational Intelligence and Neuroscience, vol. 2016, Article ID 8680541, 14 pages, 2016. doi:10.1155/2016/8680541

[8] F. Timm and E. Barth, &ldquo,Accurate Eye Centre Localisation by Means of Gradients,&rdquo,Proc. Sixth Int',l Conf. Computer Vision Theory and Applications, 2011.